#include <stdio.h>
int main(void)
{
    int count;
    for (count = 1; count <= 500; count++)
        printf("I will not throw paper airplanes in class.");
    return 0;
}
PHP syntax template

- any contents of a .php file between <?php and ?> are executed as PHP code
- all other contents are output as pure HTML
Arrays

```
$name = array();  # create
$name = array(value0, value1, ..., valueN);

$name[index]  # get element value
$name[index] = value;  # set element value
$name[] = value;  # append

$a = array();  # empty array (length 0)
$a[0] = 23;  # stores 23 at index 0 (length 1)
$a2 = array("some", "strings", "in", "an", "array");
$a2[] = "Ooh!";  # add string to end (at index 5)
```

- to append, use bracket notation without specifying an index
- element type is not specified; can mix types
### Array functions

<table>
<thead>
<tr>
<th>function name(s)</th>
<th>description</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>number of elements in the array</td>
</tr>
<tr>
<td>print_r</td>
<td>print array's contents</td>
</tr>
<tr>
<td>array_pop, array_push, array_shift, array_unshift</td>
<td>using array as a stack/queue</td>
</tr>
<tr>
<td>in_array, array_search, array_reverse, sort, rsort, shuffle</td>
<td>searching and reordering</td>
</tr>
<tr>
<td>array_fill, array_merge, array_intersect, array_diff, array_slice, range</td>
<td>creating, filling, filtering</td>
</tr>
<tr>
<td>array_sum, array_product, array_unique, array_filter, array_reduce</td>
<td>processing elements</td>
</tr>
</tbody>
</table>
• the array in PHP replaces many other collections in Java
  • list, stack, queue, set, map, ...
The foreach loop

```php
foreach ($array as $variableName) {
    ...
}
```

```php
$stooges = array("Larry", "Moe", "Curly", "Shemp");
for ($i = 0; $i < count($stooges); $i++) {
    print "Moe slaps {$stooges[$i]}\n";
}
foreach ($stooges as $stooge) {
    print "Moe slaps $stooge\n";  # even himself!
}
```

• a convenient way to loop over each element of an array without indexes
Math operations

\[ a = 3; \]
\[ b = 4; \]
\[ c = \sqrt{\text{pow}(a, 2) + \text{pow}(b, 2)}; \]

<table>
<thead>
<tr>
<th>abs</th>
<th>ceil</th>
<th>cos</th>
<th>floor</th>
<th>log</th>
<th>log10</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>min</td>
<td>pow</td>
<td>rand</td>
<td>round</td>
<td>sin</td>
<td>sqrt</td>
<td>tan</td>
</tr>
</tbody>
</table>

- the syntax for method calls, parameters, returns is the same as Java
$name = "Victoria";
$name = NULL;
if (isset($name)) {
    print "This line isn't going to be reached.\n";
}
Print/ing HTML tags in PHP = bad style

```php
<?php
print "<!DOCTYPE html>\n";
print "<html>\n";
print "  <head>\n";
print "    <title>Geneva's web page</title>\n";
...
for ($i = 1; $i <= 10; $i++) {
    print "<p class="count"> I can count to $i! </p>\n";
}
?>
```

- printing HTML tags with print statements is bad style and error-prone:
  - must quote the HTML and escape special characters, e.g. \\
- but without print, how do we insert dynamic content into the page?
## PHP expression blocks

<table>
<thead>
<tr>
<th>PHP expression block:</th>
<th>evaluates and embeds an expression's value into HTML</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;?= expression ?&gt;</code></td>
<td><code>&lt;?php print expr; ?&gt;</code></td>
</tr>
</tbody>
</table>

The answer is 42

```
<h2> The answer is `<?= 6 * 7 ?>` </h2>
```

The answer is 42

```
<h2> The answer is `<?= 6 * 7 ?>` </h2>
```

The answer is 42
Expression block example

```html
<!DOCTYPE html>
<html>
<head><title>CSE 154: Embedded PHP</title></head>
<body>
  <?php for ($i = 99; $i >= 1; $i--) { ?>
  <p><? echo $i; ?></p>
  <p><? echo $i; ?> bottles of beer on the wall, <br />
      Take one down, pass it around, <br />
      <? echo $i - 1; ?> bottles of beer on the wall. </p>
  <?php } ?>
</body>
</html>
```
Common errors: unclosed braces, missing = sign

```php
<body>
    <p>Watch how high I can count:
        <?php for ($i = 1; $i <= 10; $i++) { ?>
            $i
        <? $i ?>
    </p>
</body>
</html>
```

- `</body>` and `</html>` above are inside the for loop, which is never closed
- if you forget to close your braces, you'll see an error about 'unexpected $end'
- if you forget `=` in `<?=`, the expression does not produce any output
Complex expression blocks

```php
<?php for ($i = 1; $i <= 3; $i++) { ?>
    <h<?= $i ?>>This is a level <?= $i ?> heading.</h<?= $i ?>>
<?php } ?>
```

This is a level 1 heading.
This is a level 2 heading.
This is a level 3 heading.

- expression blocks can even go inside HTML tags and attributes
functions

function name(parameterName, ..., parameterName) {
    statements;
}

function bmi($weight, $height) {
    $result = 703 * $weight / $height / $height;
    return $result;
}

• parameter types and return types are not written
• a function with no return statements is implicitly "void"
• can be declared in any PHP block, at start/end/middle of code
Calling functions

```php
$name(expression, ..., expression);

$w = 163;  # pounds
$h = 70;   # inches
$my_bmi   = bmi($w, $h);
```

- if the wrong number of parameters are passed, it's an error
Variable scope: global and local vars

```php
$school = "UW";               # global
...

function downgrade() {
    global $school;
    $suffix = "(Wisconsin)";   # local

    $school = "$school $suffix";
    print "$school\n";
}
```

- variables declared in a function are local to that function; others are global
- if a function wants to use a global variable, it must have a global statement
  - but don't abuse this; mostly you should use parameters
Default parameter values

```php
function name(parameterName = value, ..., parameterName = value) {
    statements;
}
```

```php
function print_separated($str, $separator = "", ") {
    if (strlen($str) > 0) {
        print $str[0];
        for ($i = 1; $i < strlen($str); $i++) {
            print $separator . $str[$i];
        }
    }
}
```

```php
print_separated("hello");        # h, e, l, l, o
print_separated("hello", ")-");  # h-e-l-l-o
```

- if no value is passed, the default will be used (defaults must come last)